DevOps for IoT: A Hammer-io Extension

Team: sddec19-24 (http://sddec19-24.sd.ece.iastate.edu/)
Members: Matt Bechtel, Brett Wilhelm, Yussef Saleh, Ahmed Sobi, Chakib Ahlouche
Client/Advisor: Lotfi Ben-Othmane

- **Problem**
  - The existing Hammer-io system allows for the deployment of a user's application to a cloud service (namely Heroku), but doesn't allow for deployment to low resource IoT devices, devices that cannot perform virtualization and are assumed to be owned by the client.
- **Solution**
  - Add two new services to the Hammer-io system, the IoT Deployment Service and IoT Deployment Manager. The service will run alongside the rest of the system, in the cloud, and the manager will run on each device the user desires to deploy to. The service and the manager will work in tandem to run the clients software directly from their git repository without the use of virtualization technologies such as Docker.

Technical Details

- **Backend Technologies**
  - NodeJS
  - PostgreSQL
- **Frontend Technologies**
  - React
- **Languages**
  - Backend/Frontend written in JavaScript
- **Source Control**
  - Git

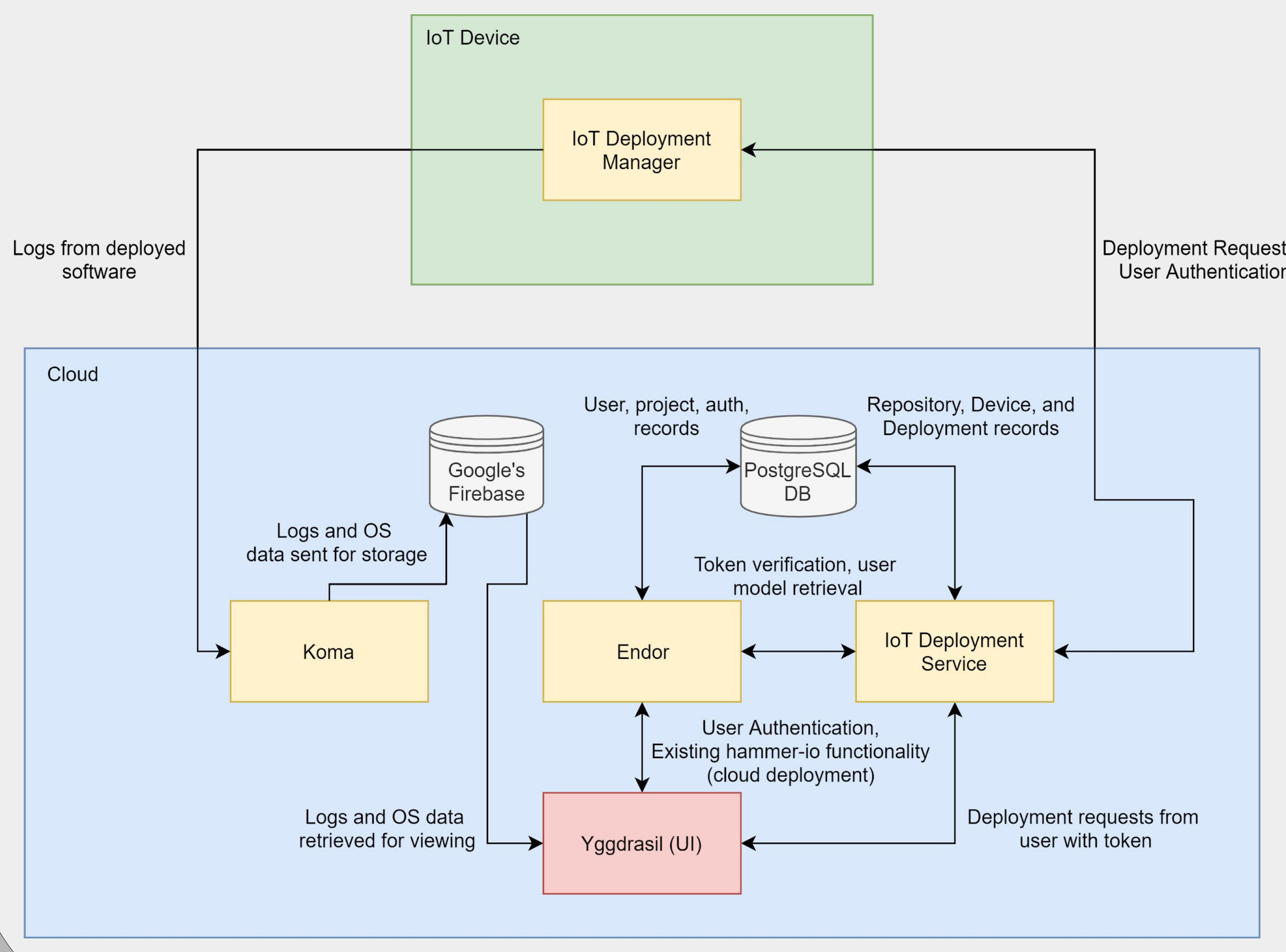
Requirements

- **Functional Requirements**
  - Extension adds IoT Deployment functionality to the Hammer-io system
    - User can deploy software from their repository on their IoT device using the Hammer-io system.
    - The implementation should utilize the existing user access functionality provided by Endor to authenticate deployment requests, making sure that only users who own a given repository or device can perform deployments with them.
    - Deployment functionality should be available via the UI or via a CI route using Git Hooks.
    - UI should be extended to provided the user easy access to all IoT deployment functionality.
- **Non-Functional Requirements**
  - API should be developer friendly with documentation for easy use (swagger docs)
  - API endpoints should require a valid user token to perform any action.
  - API should respond with correct error codes and messages
  - Hammer-io system should be able to be easily distributed via source code for each service and via docker images with an accompanying Docker Compose script (this allows for the standing up the of the cloud services via a single command and a mounted configuration file).
- **Engineering Assumptions/Constraints**
  - IoT Device
    - The IoT device is a Linux device that is connected to the internet at an address that is addressable by the IoT Deployment Service.
    - The IoT device is owned by the user and has the IoT Deployment Manager actively running on it.
    - The IoT device does not have enough resources to perform virtualization, thus the usage of Docker is unavailable.

Testing

- **Unit Testing**
  - All middleware functionality is unit tested via the Mocha framework.
- **Integration Testing**
  - All API functionality is tested using Postman tests.
- **Manual testing**
  - All functionality is tested manually alongside integration and unit tests at the time of the pull request and code review.

The Extended Hammer-io System Component Diagram


IoT Device
IoT Deployment Manager
Logs from deployed software
Deployment Requests, User Authentication
Cloud
User, project, auth, records
Repository, Device, and Deployment records
Google's Firebase
PostgreSQL DB
Logs and OS data sent for storage
Token verification, user model retrieval
Koma
Endor
IoT Deployment Service
User Authentication, Existing hammer-io functionality (cloud deployment)
Logs and OS data retrieved for viewing
Yggdrasil (UI)
Deployment requests from user with token

Component Definitions

- **Endor**
  - Existing backend web service built to handle all user management and authentication. Also handles existing cloud deployment requests, which involves the storing of a user's credentials and direct contact with cloud services such as Heroku to facilitate deployments.
- **Koma**
  - Data aggregation service for all deployed instances. Collecting log and OS data.
- **Yggdrasil** (React frontend)
  - User Interface. Surfacing all of the systems functionality in usable manner. Yggdrasil was built by the previous team and was extended upon to add pages for IoT deployment.
- **IoT Deployment Service**
  - New service adding functionality for IoT deployment. This service will be hit directly by the user (UI) to perform deployments to a given IoT device. This service will authenticate all inbound requests with the user access endpoints provided in Endor. Service will communicate with IoT Deployment Manager running on the user's IoT device to perform deployments of the user's software. The service will be responsible for storing deployment records, repository information, and device information.
- **IoT Deployment Manager**
  - The Deployment Manager will run on the user's device and receive deployment requests directly from the associated Deployment Service. The manager will authenticate all requests with the Deployment Service, ensuring that the token in the request belongs to the user who owns the device they are attempting to deploy to. Once authenticated, the Manager will execute the user provided start script within the repository to start the software. Once running, logs produced by the running software (the deployment) will be sent to Koma for aggregation.

Most Relevant Standards

- IEEE-12207: Software life cycle processes
  - This standard defines the processes to be followed throughout a piece of software's lifecycle from inception to production to maintenance.
  - This standard was important in influencing the way we went about developing our system.
- IEEE-1619: Standard for Cryptographic Protection of Data on Block-Oriented Storage Devices
  - We are storing private credentials, so we used this standard as a framework to go about keeping our user's data safe and secure.

Project Resources

- https://nodejs.org/en/docs/
- https://sequelize.org/v5/index.html
- https://www.nodegit.org/
- https://www.ieee.org/
- https://sdmay18-19.sd.ece.iastate.edu/